Emptiness check for Büchi Automata based on decomposition

Etienne Renault

LRDE/LIP6

July 5, 2012

What is Model Checking?

Check if a given system respects the specified behaviour.

We need:

- a system: a microwave owen,
- a property:
  - The owen doesn't heat up until the door is closed.
  - If start button is pressed, the owen will heat up in the future.

Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.

What is Model Checking?

Check if a given system respects the specified behaviour.

We need:

- a system: a microwave owen,
- a property:
  - The owen doesn't heat up until the door is closed.
  - If start button is pressed, the owen will heat up in the future.

Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.

What is Model Checking?

Check if a given system respects the specified behaviour.

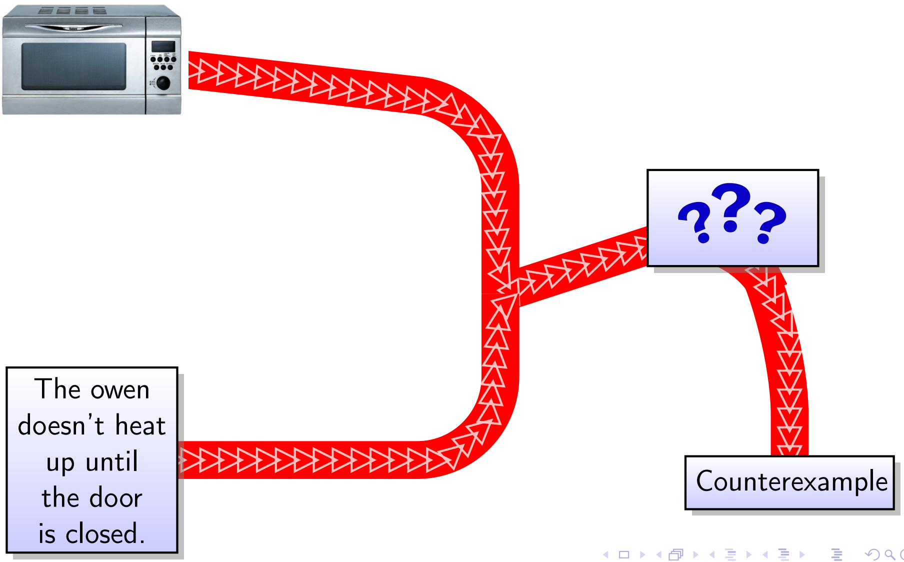
We need:

- a system: a microwave owen,
- a property:
  - The owen doesn't heat up until the door is closed.
  - If start button is pressed, the owen will heat up in the future.

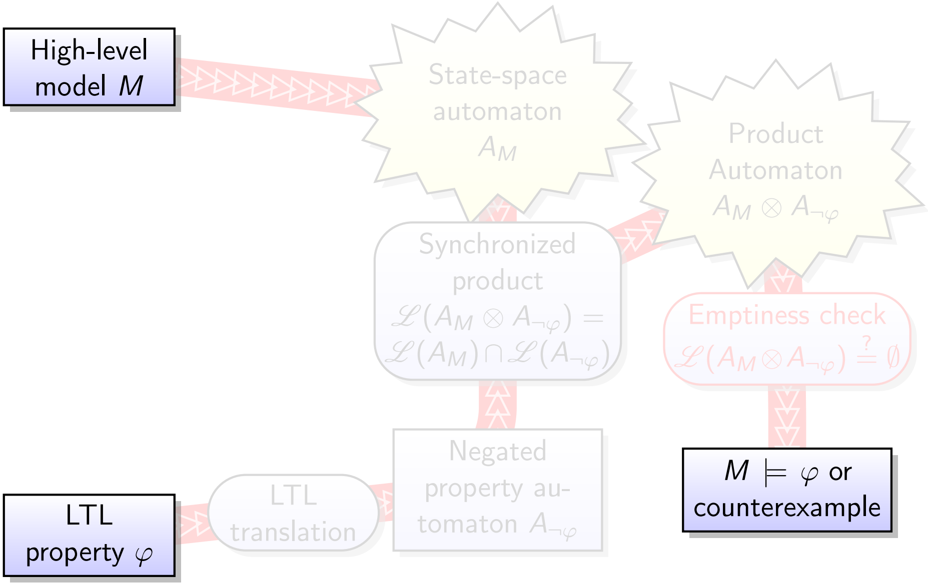
Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.

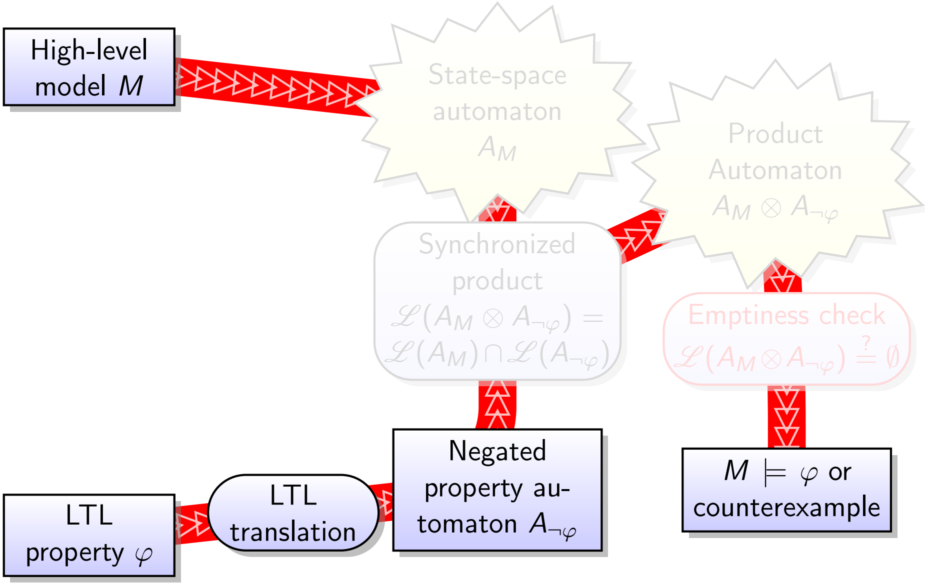
What is Model Checking?

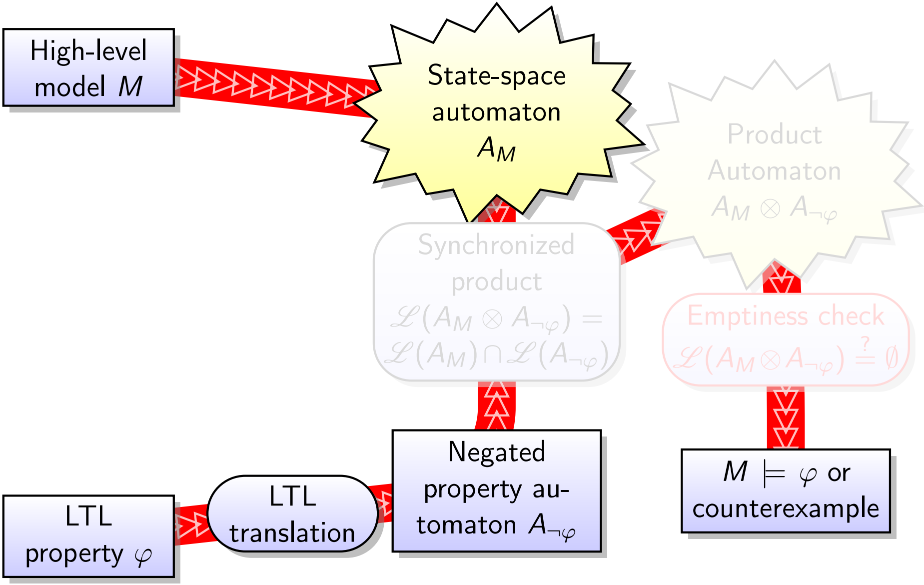


Automata-Theoretic LTL Model Checking

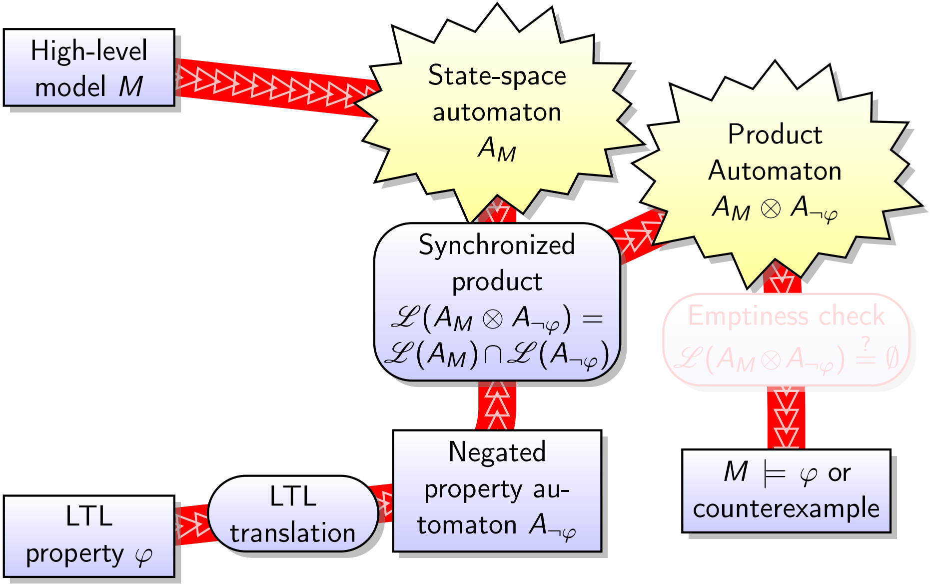


Automata-Theoretic LTL Model Checking

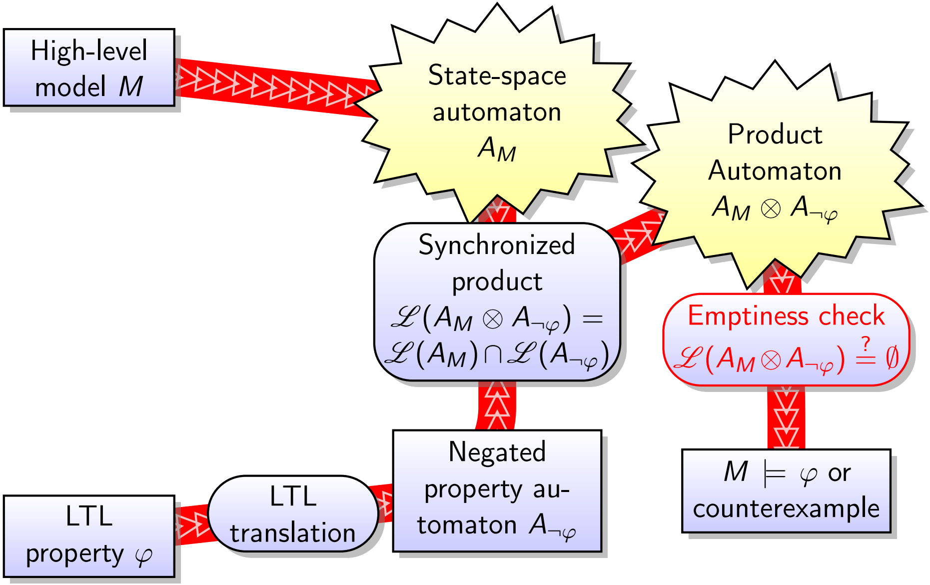



High-level model M
State-space automaton A_M
Product Automaton $A_M \otimes A_{\neg\varphi}$
Synchronized product $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$
Emptiness check $\mathscr{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$
LTL property φ
LTL translation
Negated property automaton $A_{\neg\varphi}$
$M \models \varphi$ or counterexample

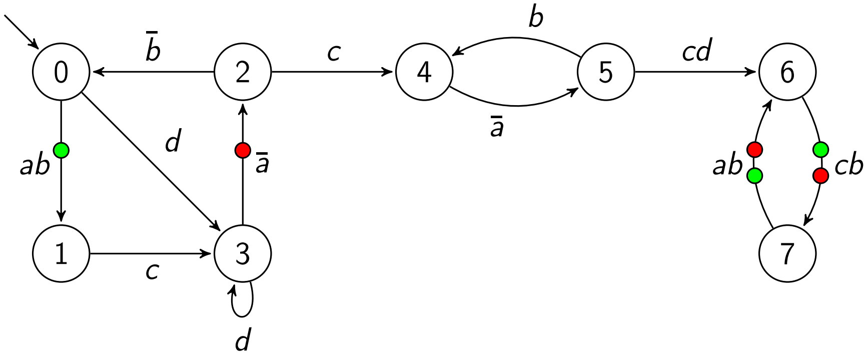
Automata-Theoretic LTL Model Checking



Automata-Theoretic LTL Model Checking

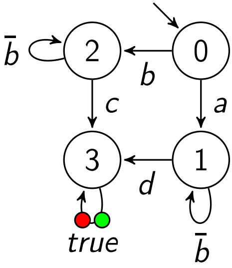
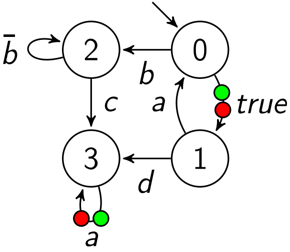
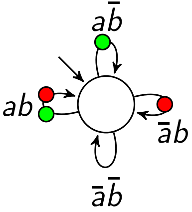


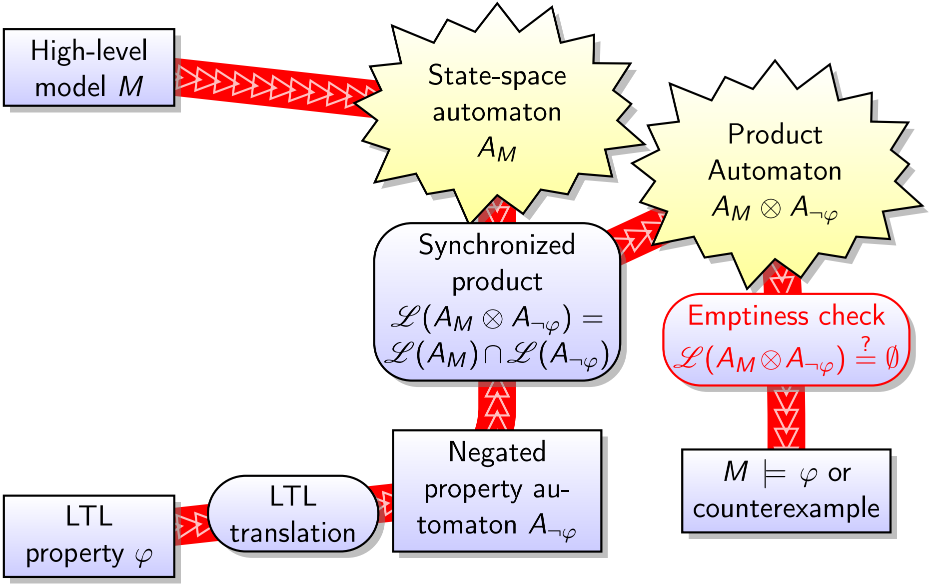
Transition-based Generalized Büchi Automaton



Accepting runs are infinite sequences visiting infinitely often each acceptance conditions

Categories of Automata

Terminal Automaton	Weak Automaton	Strong Automaton
		
Reachability	DFS	NDFS


High-level model M
State-space automaton A_M
Product Automaton $A_M \otimes A_{\neg\varphi}$
Synchronized product $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$
Emptiness check $\mathscr{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$
LTL property φ
LTL translation
Negated property automaton $A_{\neg\varphi}$
$M \models \varphi$ or counterexample

Global approach

- Approach proposed by Somenzi, Bloem and Ravi (CAV'99).
- Over-approximate class syntactically (from the formula).
- Apply the more efficient emptiness check algorithm.

Starting point

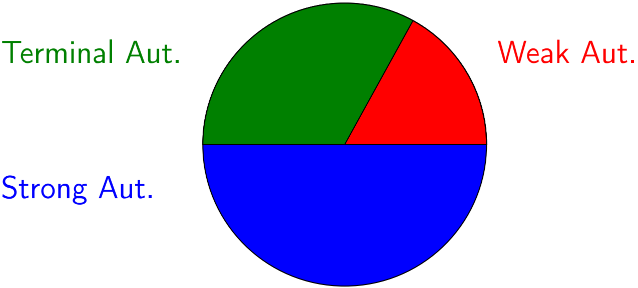
Automata can be composed of subautomata of each type, how can we use this information to perform efficient emptiness check?

Our works

- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.

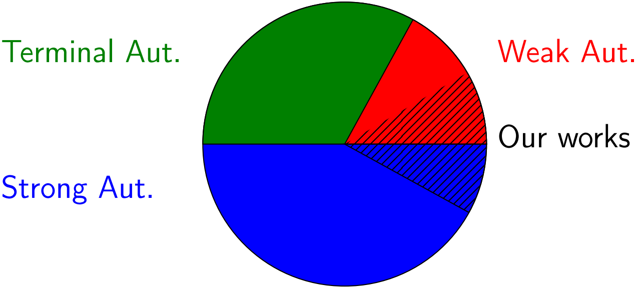
Our works

- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.

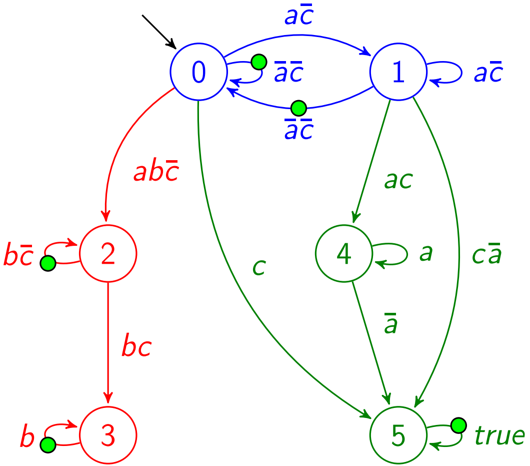


Our works

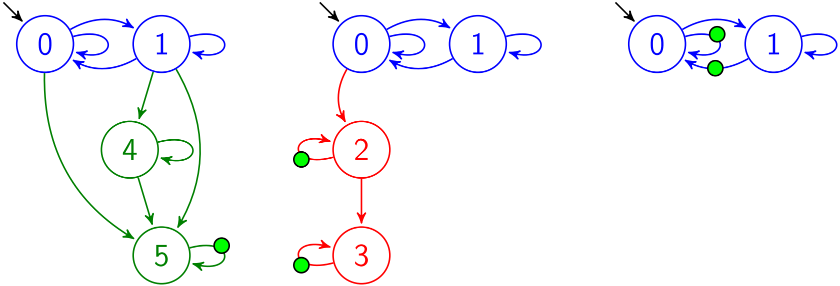
- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.



Example of Decomposition for $(\mathtt{G}\,a \rightarrow \mathtt{G}\,b)\,\mathtt{W}\,c$

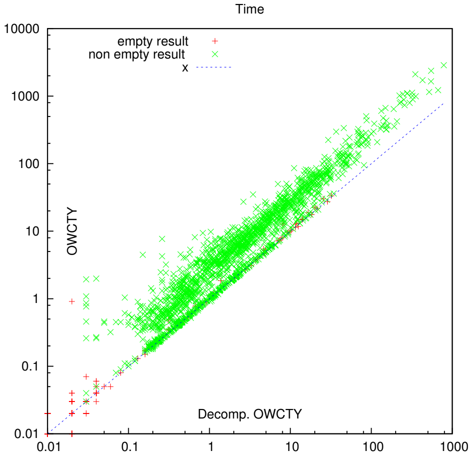


Terminal Aut.	Weak Aut.	Strong Aut.



- As soon as a counterexample is found, kill other emptiness checks.
- Otherwise, wait for the end of all emptiness checks.

Benchs



- Models: Ring, Fms, Kanban, Philo
- 2600 formulas
- 427 empty result
- 2173 counterexamples found

Conclusion and future works

- Minimising the original automaton by composing all minimized subautomata.
- Extracting other automata.
- Mixing decomposition with symbolic, explicit and hybrid approaches.
- Considering other temporal logics (PSL is already supported).
- Mixing this approach with other type of automata (Streett, testing automata,...).
- Mixing this with other techniques of verification (Partial Order, SAT,...).

That's all folks...

Questions?